KATHOLIEKE
UNIVERSITEIT
LEUVEN

DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

RESEARCH REPORT 0261

MATHEMATICAL PROGRAMMING BASED HEURISTICS FOR IMPROVING LP-GENERATED CLASSIFIERS

by
J. ADEM
W. GOCHET

D/2002/2376/61

Mathematical Programming Based Heuristics for Improving LP-generated Classifiers

Jan ADEM
Department of Applied Economics, Katholieke Universiteit Leuven,
Naamsestraat 69, 3000 Leuven, Belgium,
e-mail : jan.adem@econ.kuleuven.ac.be

Willy GOCHET
Department of Applied Economics, Katholieke Universiteit Leuven,
Naamsestraat 69, 3000 Leuven, Belgium,
e-mail : willy.gochet@econ.kuleuven.ac.be

Abstract: Mathematical programming is used as a nonparametric approach to supervised classification. However, mathematical programming formulations that minimize the number of misclassifications on the design dataset suffer from computational difficulties. We present mathematical programming based heuristics for finding classifiers with a small number of misclassifications on the design dataset. The basic idea is to improve an LP-generated classifier with respect to the number of misclassifications on the design dataset. The heuristics are evaluated computationally on both simulated and real world datasets.

Keywords: Heuristics, Supervised Classification, Mixed Integer Linear Programming.

1 Introduction

Various attempts to solve supervised classification (SC) problems with mathematical programming (MP) techniques in the past gave rise to a rich variety of MP problems. [8] gives a good overview until 1990. Most of these MP problems are linear programming (LP) problems. Their objective function is to optimise some kind of linear distance measure, which sometimes acts as a surrogate for design dataset error rate minimization. Contrary to LP problems, mixed integer linear programming (MILP) problems can directly attack the objective of minimizing the number of misclassifications on the design dataset. In the past, several researchers [13, 1, 9, 20] have tried to formulate SC problems as MILP problems. While in the 1980's and before, the attention of researchers was mainly directed at exploring different LP and MILP formulations, in the 1990's the focus point shifted towards finding good solution methods to solve these MILP problems since, in contrast to LP problems, they suffer from computational difficulties. The efforts are mainly focussed at solving efficiently the two-group SC problem. For example [16, 17, 19, 15, 2, 12] have developed solution methods for the two-group SC problem, both optimal and heuristic. As far as the optimal solution methods are concerned, it is not clear what can be expected with regard to real world datasets. However, there are a few very good MP based heuristics [6, 16] that can solve real world two-group SC problem instances fast.

The aim of our work is to develop efficient MP based heuristics for SC problems with more than two classes. In Section 2, we describe the SC problem, present a MILP formulation and discuss some properties of its LP relaxation. The optimal solution of the LP relaxation of this MILP formulation will be the starting point of our MP based heuristics. The heuristics will be described in Section 3. Section 4 gives computational results on both simulated and real world datasets. The conclusions and suggestions for future research are summarized in the conclusions.

2 Mathematical Programming Formulation

The SC problem consists of finding a formal rule that classifies patterns with unknown class membership into one of a finite number of classes C as accu-

rately as possible. If $C = 2$ the problem is called the two-group SC problem. Such a formal rule is called a classifier. Patterns are whatever needs to be classified. For each pattern the values of P measurements are known. Selection of measurements that are meaningful for class membership prediction is a difficult problem in itself and commonly referred to as feature selection and feature extraction [22]. In order to design a classifier, a design dataset consisting of a finite number of patterns N with known class membership is given. Denote by N_c the number of patterns that belong to class $c \in \{1, ..., C\}$. Let $x_{np} \in \text{IR}$ with $n \in \{1, ..., N\}$ and $p \in \{1, ..., P\}$ be the value of measurement p of pattern n and $c_n \in \{1, ..., C\}$ the class to which pattern n belongs. Overviews of the currently available techniques for designing classifiers are given in textbooks such as [22]. One way of construction is through the use of C functions of the form

$$g_c : \text{IR}^P \mapsto \text{IR} : (x_1, ..., x_P) \mapsto g_c(x_1, ..., x_P) = a_{c0} + \sum_{p=1}^{P} a_{cp} x_{np}$$

with $a_{c0}, a_{cp} \in \text{IR}$ [3, 10]. The functions are labelled such that function g_1 is associated with class 1, function g_2 with class 2 etc. This link suggests the following classifier: classify a pattern with measurements $x_1, ..., x_P$ into one of the classes c^* for which $c^* = \arg\max_c \{g_c(x_1, ..., x_P)\}$. If a classifier classifies a pattern into an incorrect class then the pattern is referred to as a misclassification. Let ϵ be a small strictly positive constant and M a sufficiently large strictly positive constant. By introducing the N binary variables $y_n = 1$ if pattern n is misclassified and $= 0$ otherwise, the classifier with the minimum number of misclassifications on the design dataset is obtained by solving the following MILP problem:

$$\text{MILP} - M\epsilon : \min_{a_{cp}, y_n} \sum_{n=1}^{N} y_n$$

subject to

$$(a_{c_n 0} + \sum_{p=1}^{P} a_{c_n p} x_{np}) - (a_{c0} + \sum_{p=1}^{P} a_{cp} x_{np}) + M y_n \geq \epsilon$$

$$n \in \{1, ..., N\} \quad c \in \{1, ..., c_n - 1, c_n + 1, ..., C\}$$

$$a_{cp} \in \text{IR} \quad c \in \{1, ..., C\} \quad p \in \{0, 1, ..., P\}$$

$$y_n \in \{0, 1\} \quad n \in \{1, ..., N\}$$

Without loss of generality, one of the functions g_c can be set to 0 [10]. Setting the right hand sight to ϵ rather than zero prohibits the trivial solution from being optimal, i.e. the solution with $a_{cp} = a_{dp}$ for $c, d \in \{1, ..., C\}$ and $p \in \{0, 1, ..., P\}$. It is easy to check that the value of ϵ only affects the optimal solution up to a multiplicative skalar and does not change the objective function value of MILP-ϵM provided M is sufficiently large. The optimal objective function value of MILP-ϵM is invariant to any linear transformation of the measurements and the optimal solution can be transformed accordingly (see Appendix A). M should be sufficiently large but no value can be given which is guaranteed to be large enough as for any value of M it is possible to construct an MILP-ϵM instance for which it is too small and cuts away the optimal solution (see Appendix B). Such instances are however unlikely to be uncountered in practice. As MILP-ϵM always has a nonempty feasible region, a classifier with the minimum number of misclassifications on the design dataset always exists.

The optimal solution of the LP relaxation of MILP-ϵM is the starting point of the MP based heuristics. The LP relaxation of MILP-ϵM can be written as

$$\min_{a_{cp}, y_n} \{\sum_{n=1}^{N} (\max_{c} \{\frac{(a_{c0} + \sum_{p=1}^{P} a_{cp} x_{np}) - (a_{c_n 0} + \sum_{p=1}^{P} a_{c_n p} x_{np}) + \epsilon}{M}\}, 0)\}$$

with $a_{cp} \in$ IR. The values of M and ϵ only scale the objective function value of the LP relaxation. The optimal solution is the same for all strictly positive values M and ϵ. The strictly positive value of ϵ does not prevent the trivial solution from being optimal in the LP relaxation. As in [3] sufficient and necessary conditions for the trivial solution to be optimal in the LP relaxation can be derived (see Appendix C) and are independent of M and ϵ. The optimal objective function value of the MILP-ϵM LP relaxation is also invariant to any linear transformation of the measurements and the optimal solution can be transformed accordingly. The feasible region of the LP relaxation is always nonempty. The LP relaxation lower bound is weak in general. The gap will only be zero when a classifier with zero misclassification on the design dataset is possible.

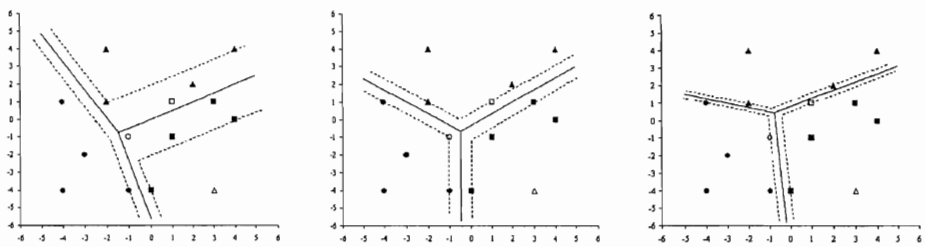
3 Mathematical Programming Based Heuristics

Solving MILP-ϵ to optimality is computationally difficult in general. LINDO quickly runs into extremely large computation times or software problems. We have tried to write a more efficient branch and bound algorithm using different branching and search strategies in the branch and bound tree. Unfortunately, there was little consistency in the computational results: which strategy works best is highly dependent on the design dataset to be solved and, in general, the computational effort needed remains high. The fundamental problem with the branch and bound idea is the lack of a good lower bound. Similar research for the two-group SC problem points to the same conclusion [17].

We have developed two sets of heuristics. The common idea behind both sets of heuristics is to improve the LP-generated classifier with respect to the number of misclassifications on the design dataset.

3.1 AG Heuristics

Sometimes it is not difficult to improve an LP-generated classifier with respect to the number of misclassifications on the design dataset. Consider the stage 1 classifier in Figure 1(a) obtained by solving the MILP-ϵM LP relaxation.



(a) stage 1 classifier (b) stage 2 classifier (c) stage 3 classifier

Figure 1: Improving the MILP-ϵM LP relaxation classifier

The stage 1 classifier has three misclassifications on the design dataset (in-

dicated by hollow objects in Figure 1(a)). The full lines indicate the classification boundary while the dashed lines show the boundaries $+\epsilon$ and $-\epsilon$. For all patterns n that are correctly classified, add the constraint $y_n = 0$ to the MILP-ϵM LP relaxation formulation. The addition of these constraints ensures that any new optimal MILP-ϵM LP relaxation solution will keep these patterns (indicated by non-hollow objects in 1(a)) correctly classified. Choose one of the patterns that are incorrectly classified, say pattern $\dot{n}$ with measurements $(-1, -1)$ and add the constraint $y_{\dot{n}} = 0$ to the MILP-ϵM LP relaxation formulation and solve it. Figure 1(b) displays the new optimal classifier, labelled stage 2 classifier. It has only two misclassifications. Again choose one of the patterns that are incorrectly classified, e.g. pattern $\dot{n}$ with measurements $(1, 1)$. Add the constraint $y_{\dot{n}} = 0$ and solve the MILP-ϵM LP relaxation formulation which yields the stage 3 classifier shown in Figure 1(c). The stage 3 classifier is an optimal solution to the MILP-ϵM instance.

Iteratively applying this idea leads to a heuristic for improving LP-generated classifiers with respect to the number of misclassifications on the design dataset. A formalized description of the heuristic is given below.

denote by $\mathbf{X} \in \mathrm{IR}^{N \times P}$ the matrix of measurements of the design datset patterns. Let $(a^*_{c0}, \mathbf{a}^*_c, \mathbf{y}^*)$ with $\mathbf{a}^*_c = [a^*_{c1} ... a^*_{cP}]$ and $\mathbf{y}^* = [y^*_1 ... y^*_N]$ with $c \in \{1, ..., C\}$ and $n \in \{1, ..., N\}$ represent the optimal solution of the MILP-ϵM LP relaxation.

input: $N, C, P, \mathbf{X}$
initialize: $ub_{best} = N$
$solution_{best} = (0, \mathbf{0}, \mathbf{0})$
$level = 0$
solve MILP-ϵM LP-relaxation
step 1: $level \leftarrow level + 1$
$solution_{current} = (a^*_{c0}, \mathbf{a}^*_c, \mathbf{y}^*)$
$F_{level} = \{n \mid c_n \neq \arg\max_c \{g_c(x_{n1}, ..., x_{nP})\}\}$
$ub_{current} = |F_{level}|$
if $ub_{current} < ub_{best}$ then
$ub_{best} \leftarrow ub_{current}$
$solution_{best} \leftarrow solution_{current}$
goto step 2
step 2: if $|F_{level}| = 0$, goto step 6, else goto step 3.
step 3: choose $\dot{n} \in F_{level}$
change $F_{level} \leftarrow F_{level} \setminus \{\dot{n}\}$
add constraints $y_n = 0$ for $n \notin F_{level}$
add constraint $y_{\dot{n}} = 0$
solve MILP-ϵM LP relaxation

goto step 4
step 4: if feasible, goto step 1, else goto step 5
step 5: remove constraint $y_{\dot{n}} = 0$, goto step 2
step 6: stop
output: ub_{best}
$solution_{best}$

Only in the initialization, it is possible to obtain the trivial optimal solution. This solution sets $ub_{current} = N$ but does not jeopardize the correct functioning of the heuristic. There are several possible strategies for choosing a pattern $\dot{n} \in F_{level}$ in step 3. These strategies are expected to determine the computational efforts of the heuristic and the quality of the upper bound. A strategy that is guaranteed to find the MILP-ϵM optimum cannot be found.

AG-y
: Select a pattern $\dot{n} = \arg\min_n\{y_n | n \in F_{level}\}$. A small y_n value indicates that pattern n is close to being classified correctly. Given the correctly classified patterns, it should be relatively easy to classify this pattern also correctly.

AG-stLP
: Calculate for each pattern $\tilde{n} \in F_{level}$ the MILP-ϵM LP optimum $z_{\tilde{n}}$ after adding the constraints $y_n = 0$ for $n \notin F_{level}$ and $y_{\tilde{n}} = 0$. Select a pattern $\dot{n} = \arg\min_{\tilde{n}}\{z_{\tilde{n}} | \tilde{n} \in F_{level}\}$. The number of elements in F_{level} determines the number of extra LP problems to be solved and strongly influences the computation time.

AG-stIP
: The same as in AG-stLP but now based on $ub_{\tilde{n}}$ values in stead of $z_{\tilde{n}}$ where $ub_{\tilde{n}}$ is the value of the upper bound associated with the optimal MILP-ϵM LP solution.

AG-stIPs
: Calculate for each pattern $\tilde{n}$ in F_{level} the set $S_{\tilde{n}} = \{n | y_n = 0\}$ and the upper bound $ub_{\tilde{n}}$ in the MILP-ϵM LP optimum after adding the constraints $y_n = 0$ for $n \notin F_{level}$ and $y_{\tilde{n}} = 0$. Select those patterns $\tilde{n}$ for which $S_{\tilde{n}}$ is no subset of any other set $S_{\breve{n}}$ with $\breve{n} \in F_{level}$ and $\breve{n} \neq \tilde{n}$. Backtracking is done only on these patterns $\tilde{n}$. As in AG-stIP, we first select a pattern $\dot{n} = \arg\min_{\tilde{n}}\{ub_{\tilde{n}} | \tilde{n} \in F_{level}\}$.

Backtracking to the previous *level* can be allowed for by changing step 6 of the AG heuristic.

step 6: remove all constraints $y_n = 0$
$level \leftarrow level - 1$
if $level = 0$, then stop, else goto step 2.

Even with backtracking, the heuristic is not guaranteed to find the MILP-ϵM optimum. Backtracking on all elements of F_{level} is to be avoided since it might be very time consuming (see Section 4). We choose to backtrack only on the k most promising elements of F_{level}, if there exist k elements. The procedure above corresponds to $k = 1$. We have implemented $k = 2$ and $k = 3$. A strategy for choosing $\dot{n} \in F_{level}$ only makes sense if we do not backtrack on all elements of F_{level}.

In the course of the algorithm, we continuously add and remove constraints to the MILP-ϵM LP relaxation. In general, it is more efficient to solve the dual of the MILP-ϵM LP relaxation since it has a better computational structure than the primal. Adding or removing primal constraints of the type $y_n = 0$ comes down to adding and removing free variables with the dual objective function coefficient 0 in the dual. The cost of updating the optimal LP solution in the dual after such addition or removal is relatively small because reoptimisation starts from the previous optimal solution which usually gives a good starting basis such that only a small number of simplex iterations are needed.

3.2 CH Heuristics

Chinneck [6] developed polynomial time heuristics for the maximum feasible subsystem (MAX FS) problem. The idea of the heuristics is to iteratively eliminate constraints from the infeasible LP problem until a feasible LP problem is obtained. The number of remaining constraints in this feasible LP gives an upper bound for the MAX FS problem. The heuristics differ in the way they determine which constraint is to be removed. The removal is permanent. Backtracking on the constraint removal is not done which makes the procedures relatively fast but heuristic. A formal description of the heuristics can be found in [6]. The heuristics can be readily applied to the two-group SC problem. The empirical results in [6] indicate that the classifiers have a high accuracy on the design dataset and are found fast.

It is straightforward to extend these heuristics to SC problems with more than two classes. Observe that in the two-group SC problem, for each pattern exactly one constraint is in the MILP-ϵM formulation and that the y_n variables in the MILP-ϵM LP relaxation have the same function as the elas-

tic variables in the elasticised version of the infeasible LP in the MAX FS problem. Iteratively removing constraints from the LP relaxation then comes down to iteratively removing patterns. The same idea, iteratively removing patterns, can also be used for the SC problem with more than two classes. Removing one pattern corresponds to deleting $C-1$ constraints. The function of the y_n variables in the MILP-ϵM LP relaxation remains unchanged, though now each y_n corresponds to $C-1$ constraints. Each time the MILP-ϵM LP relaxation is solved, the corresponding MILP-ϵM upper bound is calculated (see step 1 of description 3.1) and the classifier with the lowest upper bound value is saved as the final solution. This upper bound value might be smaller than the number of patterns that has to be removed in order to have a MILP-ϵM LP relaxation with perfect separation as nothing guarantees that once a pattern is removed, it will always be misclassified.

Several strategies are possible for deciding on which pattern is to be removed. This step of the procedure is expected to determine the speed and the quality of the upper bound. Again, one cannot find a strategy that is guaranteed to find the MILP-ϵM optimum.

CH-y
: Select a pattern $\dot{n} = \arg\max_n\{y_n\}$. A large y_n value indicates that pattern n is far from being classified correctly. In this strategy, patterns that are difficult to classify correctly are removed first.

CH-$dp.y$
: Select a pattern $\dot{n} = \arg\max_n\{dp_{ny_n}\}$ where dp_n is the largest dual price of the $C-1$ constraints associated with pattern n. $dp_n y_n$ is a good estimator for the reduction in the MILP-ϵM LP objective function when the $C-1$ constraints are left out of the formulation and we choose the pattern n that gives the strongest estimated reduction.

CH-stLP
: Make a set of candidate patterns for removal S_R. Calculate for each pattern $n \in S_R$ the MILP-ϵM LP optimum z_n with that pattern removed. Select a pattern $\dot{n} = \arg\min_n\{z_n \mid n \in S_R\}$. This strategy looks for the pattern n that gives the steepest descent in the MILP-ϵM LP optimum value. The number of elements in the set S_R determines the number of extra LP's to be solved and determines the computation time. As in [6], there are several strategies to build the set S_R. Set $S = \{n \mid y_n > 0\}$. We have implemented the following choices for S_R.

  CH - stLP(2): S_R = the elements of S with the 2 largest y_n values

  CH - stLP(3): S_R = the elements of S with the 3 largest y_n values

CH - stLP(all): $S_R = S$

CH-stIP

The same as in CH-stLP but now based on ub_n values in stead of z_n where ub_n is the value of the upper bound associated with the optimal MILP-ϵM.

Again, it is computationally more efficient to work on the dual. Removing the $C-1$ constraints of pattern n in the primal is equivalent to changing the right hand side coefficient of these constraints to $-M$. Hence, in the dual, we only have to set the $C-1$ objective function coefficients of the variables corresponding to these $C-1$ constraints to $-M$. Relatively few simplex iterations are required to reoptimise this dual LP since the previous optimal solution usually gives a good starting basis.

4 Computational results

The heuristics are tested on both simulated and real world datasets. The aim of the tests on the simulated design datasets is to compare the performance of the heuristics under varying design dataset conditions. The questions of interest are: does the heuristic find a good solution? and how fast does it find this solution? In [12], the question is raised whether or not there is a need to turn to MILP formulations for solving the SC problem. The results of previous studies by [1, 12, 20, 16, 4] are mixed. It is still unclear under which dataset conditions the MILP classifiers perform better on unseen data compared to other approaches for solving SC problems. Our best performing heuristics will be selected and ran on real world datasets to compare their performance to the performance of existing parametric and nonparametric methods.

4.1 Simulated Design Datasets

For each triplet (N, P, C) with $N \in \{25, 50, 75, 100, 250, 500, 750, 1000\}$, $P \in \{2, 3, 4, 5, 10\}$ and $C \in \{2, 3, 4, 5\}$, 10 design datasets were simulated. Patterns in each of the 1600 simulated design dataset are drawn from one of C multivariate normal distributions with covariance matrices equal to the identity matrix and the C mean vectors randomly drawn from the space

$\{0,1,2,3,4,5,6\}^P$. This space allows for a sufficient amount of variation in the overlap between the different classes in one design dataset. To generate a pattern, a random number i from $\{1,...,C\}$ is drawn and values x_1, x_2,..., x_P are taken from the associated multivariate normal distribution. Having done this N times, it is checked if each class is represented, i.e. $N_c > 0$ for $c \in \{1,...,C\}$. If this is not the case, the N patterns are rejected and resampled. Otherwise, the design dataset is accepted.

The MP based heuristics are implemented in C and make use of the LINDO solver. The experiment was run on a PENTIUM III 550 Mhz computer. The LP generated classifier is the starting solution for all procedures and took on average 0.4 seconds to determine. The 1600 design datasets are divided into four groups as their LP relaxation upper bound falls in]0,10],]10,50],]50,100] or]100,1000]. These intervals represent the difficulty of the design dataset instances. 667 design datasets are perfectly separable.

Table 1 gives statistics on the improvement on the LP-generated classifiers achieved by the different procedures. *tot* is the total number of misclassifications of the MP based heuristic solutions on all 1600 simulated design datasets expressed as a percentage of the number of misclassifications of the LP relaxation solutions. $tot]i,j]$ gives similar percentages but only for the simulated design datasets with an LP relaxation upper bound in $]i,j]$. Between brackets is the number of times the procedure yielded the lowest upper bound out of the total number of instances in this group, which is given between brackets in the top row. The average CPU time is given in hh:mm:ss format. A dash (-) indicates that we interrupted the heuristic because of extremely large CPU times on some of the design datasets and hence could not calculate the corresponding statistic. The best improvement for each group of instances is printed in bold.

All the heuristics are able to improve the LP relaxation starting solution considerably for all difficulties of the instances. Surprisingly, search strategies don't matter much. The upper bounds obtained by the more elaborated search strategies (AG-stLP, AG-stIP, AG-stLPs, CH-stLP, CH-stIP) are not significantly better than the upper bounds obtained by the simpler ones (AG-y, CH-y, CH-$dp.y$), though they are definitely more time consuming to calculate. Based on the number of times each procedure yields the lowest upper bound, the CH heuristics outperform the AG heuristics especially if

Table 1: Results on Simulated Datasets

heuristic	*tot* (933)	*tot*]0, 10] (246)	*tot*]10, 50] (332)	*tot*]50, 100] (101)	*tot*]100, 1000] (254)
LP relaxation	100% (40)	100% (40)	100% (0)	100% (0)	100% (0)
	00:00:01	00:00:01	00:00:01	00:00:01	00:00:02
AG-y(1)	49.1% (454)	41.4% (233)	41.7% (185)	46.0% (17)	50.5% (19)
	00:00:03	00:00:01	00:00:01	00:00:03	00:00:16
AG-stLP(1)	48.8% (469)	41.1% (234)	41.3% (188)	45.3% (26)	50.2% (21)
	00:01:19	00:00:01	00:00:04	00:00:29	00:06:57
AG-stIP(1)	51.0% (385)	41.7% (228)	43.5% (143)	48.1% (8)	52.3% (6)
	00:00:25	00:00:01	00:00:02	00:00:09	00:02:34
AG-stIPs(1)	50.4% (418)	41.7% (227)	42.3% (171)	47.3% (13)	51.8% (7)
	00:00:35	00:00:04	00:00:06	00:00:19	00:03:10
AG-stIPs(2)	48.8% (546)	40.3% (243)	40.2% (250)	45.1% (29)	50.4% (24)
	00:00:41	00:00:04	00:00:07	00:00:26	00:03:43
AG-stIPs(3)	48.1% (621)	**40.1%** (246)	39.5% (282)	44.2% (46)	49.8% (47)
	00:00:53	00:00:04	00:00:10	00:00:42	00:04:53
AG-stIPs (all)	-	**40.1%** (246)	**38.8%** (321)	-	-
	-	00:00:04	00:09:00	-	-
CH-y	47.0% (642)	41.9% (226)	40.7% (209)	43.7% (55)	48.2% (152)
	00:00:25	00:00:01	00:00:01	00:00:09	00:02:33
CH-*dp.y*	47.0% (641)	42.1% (224)	40.8% (209)	43.7% (54)	48.2% (154)
	00:00:25	00:00:01	00:00:01	00:00:09	00:02:31
CH-stLP(2)	**46.5%** (697)	40.8% (238)	40.2% (234)	43.6% (61)	**47.6%** (164)
	00:02:37	00:00:01	00:00:09	00:01:44	00:15:37
CH-stLP(3)	-	40.4% (242)	40.0% (242)	**43.5%** (68)	-
	-	00:00:01	00:00:18	00:03:14	-
CH-stLP(all)	-	40.2% (245)	39.8% (256)	-	-
	-	00:00:03	00:07:41	-	-
CH-stIP(2)	46.6% (653)	41.3% (233)	40.5% (217)	43.8% (52)	47.7% (151)
	00:02:40	00:00:01	00:00:08	00:01:40	00:15:54
CH-stIP(3)	-	40.7% (239)	40.4% (216)	43.8% (53)	-
	-	00:00:01	00:00:18	00:03:46	-
CH-stIP(all)	-	41.2% (234)	43.1% (125)	-	-
	-	00:00:03	00:08:38	-	-

the design dataset gets more difficult to solve. CPU time goes up quickly if the design dataset gets more difficult. AG-y(1) is the fastest heuristic. Note that almost all the CPU time in the iterations of each heuristic (e.g. for AG-y(1) 99%) is absorbed by the LP solver, here the simplex solver of LINDO. For the AG-y, AG-stLP and AG-stIP heuristics, backtracking does not result in large improvements in the upper bounds. The upper bound found after backtracking cannot be worse but computational results on the easy instances in table 2 indicate that on average the upper bound improvement is small and obtained at the cost of a large increase in CPU time. Additional tests on more difficult instances confirm this conclusion. Therefore, we will not consider the heuristics AG-y(2), AG-y(3), AG-y(all), AG-stLP(2), AG-

stLP(3), AG-stLP(all), AG-stIP(2), AG-stIP(3) and AG-stIP(all) further.

Table 2: Results of Backtracking for Easy Instances

heuristic	$tot]0,10]$	heuristic	$tot]0,10]$	heuristic	$tot]0,10]$
AG-y(1)	41.4%	AG-stLP(1)	41.1%	AG-stIP(1)	41.7%
	00:00:01		00:00:01		00:00:01
AG-y(2)	40.2%	AG-stLP(2)	40.8%	AG-stIP(2)	40.4%
	00:00:01		00:00:01		00:00:01
AG-y(3)	40.2%	AG-stLP(3)	40.7%	AG-stIP(3)	40.2%
	00:00:02		00:00:09		00:00:01
AG-y(all)	40.2%	AG-stLP(all)	40.6%	AG-stIP(all)	40.2%
	00:00:06		00:07:11		00:02:36

An interesting question is how many times we hit the global optimum with each heuristic. 667 design datasets are perfectly separable and LINDO was only able to solve 147 of the remaining 933 simulated datasets to optimality. The maximal pivot limit was hereby set to 1.000.000, allowing for a substantial amount of CPU time. Table 3 gives the number of times the MILP-ϵM optimum was found by the MP-based heuristic out of the number of times LINDO found the MILP-ϵM optimum. For 58 design datasets the MILP-ϵM optimum was 1, for 25 design datasets it was 2 etc.

Table 3: Number of Times Each MP Based Heuristic Hits the Global Optimum

	1	2	3	4	5	6	7	8	9	>9	*tot*
MILP	58	25	10	15	12	6	6	5	3	7	147
LP relaxation	27	1	0	0	0	0	0	0	0	0	28
AG-y (1)	57	23	7	13	8	4	3	5	0	4	120
AG-stLP (1)	57	22	9	12	8	3	3	5	1	4	124
AG-stIP (1)	58	21	8	9	8	3	2	2	1	1	113
AG-stIPs (1)	58	21	8	11	8	5	3	2	1	3	120
AG-stIPs (2)	58	24	10	14	9	6	4	4	1	6	136
AG-stIPs (3)	58	24	10	15	10	6	4	5	2	6	140
AG-stIPs (all)	58	24	10	15	10	6	5	5	2	7	142
CH-y	56	20	7	13	9	3	3	4	1	6	122
CH-$dp.y$	56	20	6	12	9	3	3	4	1	6	120
CH-stLP (2)	56	23	10	14	10	4	3	5	1	6	132
CH-stLP (3)	57	23	10	15	11	4	3	4	1	6	134
CH-stLP (all)	58	23	10	15	11	5	3	4	1	7	137
CH-stIP (2)	56	21	8	14	9	4	3	5	2	5	127
CH-stIP (3)	57	22	8	15	10	4	3	5	1	6	131
CH-stIP (all)	57	22	9	10	8	5	3	1	0	1	116

Search strategy does not seem to have a large impact on the ability to hit the MILP-ϵM optimum. On the easy design datasets, all heuristics obtain the MILP-ϵM optimum a large number of times. On the difficult instances, the results vary. Again the CH heuristics do slightly better than the AG heuristics. It would be interesting to see if this still holds for the difficult designs datasets (MILP-ϵM optimum $\gg$ 10).

Which of the above MP-based heuristics is best? Because there are no theoretical arguments to answer this question and the improvements of the LP generated classifier obtained by all heuristics are very similar, the main criterion to select the set of best heuristics is CPU time. Hence, the AG-stIPs(3), AG-stIPs(all), CH-stLP(all) and CH-stIP(all) heuristics will not be considered further. All other heuristics will be tested on the real world datasets for comparison with other parametric and nonparametric methods.

4.2 Real World Datasets

Datasets are available at the UCI repository at http:/kdd.ics.uci.edu. [14] use fifteen datasets from the UCI repository to test the performance of thirty-three procedures for SC. Their ten-fold cross-validation procedure will be followed in order to allow for comparison of their results with ours. Since they did not give the subsets obtained by stratified sampling on the UCI datasets, we probably did not run our procedures on the exact same subsets. Although efforts are made to level the different measures of CPU time (see Appendix D), the CPU time comparison should be seen as indicative at best.

Table 4: Characteristics of Real World Datasets

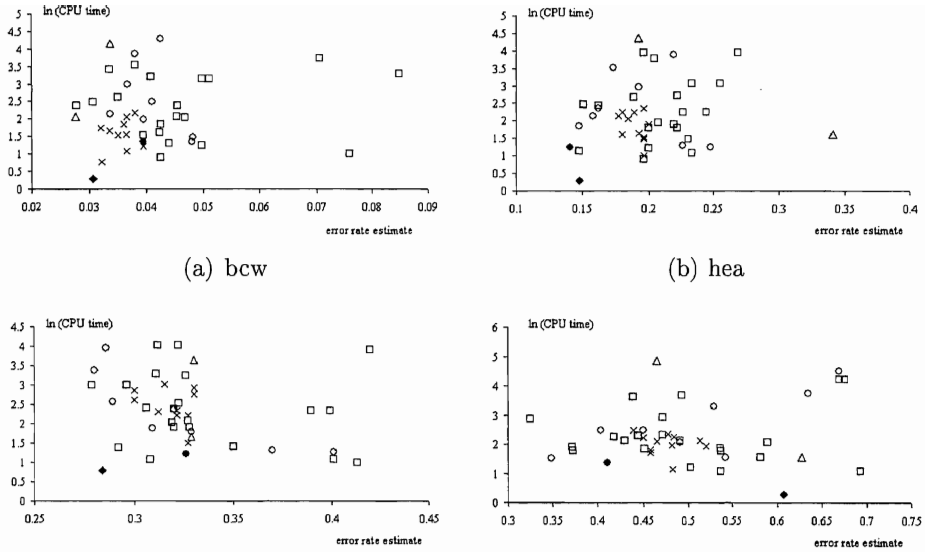
dataset	N	P	C	$(N_1;...;N_C)$
bcw	638	9	2	(444;239)
hea	270	13	2	(150;120)
bld	345	6	2	(145;200)
tae	151	5	3	(49;50;52)
ttt	958	9	2	(626;332)
ion	350	34	2	(126;224)

Only the first four datasets are also used by [14]. The last two datasets are taken because of their non-linear character and are used in a study by [21].

Table 5: Results on Real World datasets

	bcw	hea	bld	tae	ttt	ion
LP relaxation	**3.1%**	**14.8%**	**28.4%**	60.8%	33.1%	12.9%
	00:00:01	00:00:01	00:00:03	00:00:01	00:00:08	00:00:03
AG-y(1)	3.2%	19.6%	32.7%	48.2%	**15.8%**	**12.3%**
	00:00:03	00:00:05	00:00:16	00:00:07	00:02:24	00:00:09
AG-stLP(1)	3.2%	19.6%	32.7%	48.2%	**15.8%**	**12.3%**
	00:00:03	00:00:05	00:00:16	00:00:07	00:02:24	00:00:09
AG-stIP(1)	3.9%	18.1%	32.7%	48.1%	**15.8%**	14.0%
	00:00:08	00:00:20	00:01:20	00:00:46	00:08:09	00:00:25
AG-stIPs(1)	3.7%	19.3%	32.1%	52.0%	**15.8%**	14.3%
	00:00:58	00:00:22	00:01:21	00:00:43	00:09:16	00:00:31
AG-stIPs(2)	3.8%	20.0%	32.1%	51.3%	**15.8%**	15.1%
	00:01:16	00:00:40	00:01:46	00:01:05	00:14:40	00:01:17
CH-y	3.7%	19.6%	31.2%	45.7%	22.8%	13.8%
	00:00:06	00:00:17	00:01:40	00:00:32	00:27:33	00:00:12
CH-*dpy*	3.7%	19.6%	32.1%	45.7%	**15.8%**	13.8%
	00:00:06	00:00:15	00:01:21	00:00:26	00:20:56	00:00:10
CH-stLP(2)	3.7%	17.8%	30.0%	45.0%	16.5%	14.6%
	00:00:18	00:01:10	00:05:53	00:01:22	02:07:33	00:01:31
CH-stLP(3)	3.6%	19.6%	31.5%	47.7%	**15.8%**	14.0%
	00:00:36	00:01:53	00:08:48	00:01:48	04:22:29	00:02:31
CH-stIP(2)	3.5%	18.5%	33.0%	46.4%	16.3%	14.0%
	00:00:17	00:00:57	00:04:44	00:01:04	02:06:39	00:01:16
CH-stIP(3)	3.2%	18.1%	33.0%	48.3%	**15.8%**	14.0%
	00:00:28	00:01:30	00:07:09	00:01:28	03:40:03	00:02:06

A brief description of each dataset can be found at the UCI repository. The six datasets are small to moderate sized. The measurements in each dataset were standardized. For larger and more difficult datasets, the LINDO simplex solver suffers from long computation times. The experiment was carried out on a PENTIUM III 550 Mhz computer. The cross-validation error rate estimates and the calculation times in hh:mm:ss format are given in Table 5. The best error rates per dataset are indicated in bold.



(a) bcw (b) hea

(c) bld (d) tae

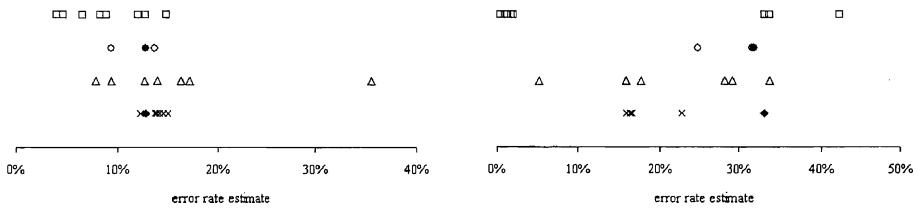
Figure 2: Results on Real World Datasets

For the four first datasets, in Figure 2 scatter plots of the natural logarithm of the calculation time expressed in seconds versus the cross-validation error rate estimate are presented. The results for the statistical procedures (indicated by $\circ$), the results for the decision trees (indicated by $\square$) and those for the neural networks (indicated by $\triangle$) can be found in [14]. Our procedures are indicated by $\times$'s. The LP relaxation classifier is indicated by $\blacklozenge$, the LDA classifier by $\bullet$.

The results are surprising. Except for the tae dataset, the LP relaxation classifier outperforms the MP based heuristics classifiers on unseen patterns in terms of error rate. Still, for each of the ten subsets of each dataset, the MP based heuristics improved the LP-generated classifier considerably. On the tae dataset, all the MP based heuristics outperform the LP relaxation classifier in terms of error rate. The LDA classifiers use the same mathematical structure to separate the classes as our classifiers. It is again surprising to see that, in terms of error rates, for the hea and tae dataset the LDA classifiers perform better than the MP based heuristics. [20] report similar

findings for their experiments: on the design datasets their MILP classifiers were significantly better in terms of error rate than both the LP and the LDA classifier while on unseen patterns the MILP classifiers performed worse than the LP and LDA classifier, except for one data setting. The reason for this poor generalized performance is probably overfitting: the logic of the MP based heuristics (and of error rate minimizing MILP formulations as such) is to adjust to design dataset peculiarities. Of course, if overfitting is so easily encountered, the usefulness of MILP formulations for error rate minimization in SC problems is put into doubt.

As to the last two datasets, our results can be compared to those of [21] although they used a slightly different method to estimate the error rate. They tested eighteen procedures on several UCI datasets. Unfortunately, they do not report calculation times. In Figure 3(a) and 3(b), plots are constructed using the symbol $\circ$ for statistical procedures, $\square$ for LS-SVM methods and $\triangle$ for the other methods used in [21]. The other procedures are indicated as in Figure 2.



(a) ion (b) ttt

Figure 3: Results on Real World Datasets

On both datasets, the non-linear classifiers produce the lowest error rate estimates. This confirms the non-linear character of the datasets. The performance of our procedures is average. On the ion dataset, both the LP relaxation classifier and the LDA classifier perform very similar to the MP

based heuristic classifiers. On the ttt dataset however, the MP based heuristic classifiers all outperform the LP relaxation classifier and the LDA classifier. Hence, also on these non-linear datasets the results are mixed.

No approach for SC is superior to all other approaches on all dataset settings. If one approach outperforms another approach on a particular dataset setting, it is a consequence of its fit to the particular instance, and not of the superiority of the approach [7]. Given an approach for SC problems, the question remains what characteristics of the design dataset will ensure that this approach will do better than other approaches. This essentially empirical question is for MILP formulations yet to be answered.

5 Conclusions

Mathematical programming formulations that minimize the number of misclassifications on the design dataset are hard to solve to optimality. The mathematical programming based heuristics we have brought forward present an alternative to the optimal solution methods. Computational experiments show that they are fast and obtain solutions often close to the MILP optimum. However, the classifiers found easily overfit the design dataset. The advantage of the mathematical programming approach in the context of supervised classification lies in its the power to model more complex real world supervised classification problems. Exploring such possibilities is part of future research.

A Appendix

Denote by $\mathbf{X}_c \in \mathrm{IR}^{N_c \mathrm{x} P}$ the matrix of measurements of patterns from class c. Let $\mathbf{U} \in \mathrm{IR}^{P \mathrm{x} P}$ be a nonsingular matrix and $\mathbf{u} \in \mathrm{IR}^{1 \mathrm{x} P}$ a vector. $\mathbf{e}$ is a column vector of ones of appropriate dimension. Denote by MILP^{tr}-ϵM the MILP-ϵM formulation with measurement vectors $\mathbf{X}_c^{tr} = \mathbf{X}_c\mathbf{U} + \mathbf{eu}$. The optimal solution to MILP-ϵM is given by $(a_{c0}^*, \mathbf{a}_c^*)$ with $\mathbf{a}_c^* = [a_{c1}^* ... a_{cP}^*]^{\mathrm{T}}$ and $c \in \{1, ..., C\}$ with objective function value z^*. Then, an optimal solution to MILP^{tr}-ϵM is $(a_{c0}^* - \mathbf{u}\mathbf{U}^{-1}\mathbf{a}_c^*, \mathbf{U}^{-1}\mathbf{a}_c^*)$ with $c \in \{1, ..., C\}$ with the same

objective function value z^*.

Proof: Assume there exists a solution $(\tilde{a}^*_{c0}, \tilde{\mathbf{a}}^*_c)$ with $c \in \{1, ..., C\}$ to MILP^{tr}-ϵM with objective function value $\tilde{z}^* < z^*$. Given a sufficiently large value of M, $(\tilde{a}^*_{c0} - \mathbf{u}\mathbf{U}^{-1}\tilde{\mathbf{a}}^*_c, \mathbf{U}^{-1}\tilde{\mathbf{a}}^*_c)$ is a feasible solution to MILP-ϵM with objective function value $\tilde{z}^* < z^*$. This contradicts that $(a^*_{c0}, \mathbf{a}^*_c)$ with $c \in \{1, ..., C\}$ is the optimal solution to MILP-ϵM. (QED)

B Appendix

Take $(N, P, C) = (7, 1, 2)$. Patterns 1, 2, 3 and 4 belong to class 1 (represented by $\triangle$), patterns 5, 6 and 7 to class 2 (represented by $\square$). The measurements are $x_{11} = 1, x_{21} = 2, x_{31} = 3, x_{41} = 15, x_{51} = 4, x_{61} = 5$ and $x_{71} = 6$ and are shown in Figure 4. Set $\epsilon = 1$. It is easy to check that for $M \leq 2(x_{41} - 3.5) + \epsilon = 24$ the optimal objective function value of MILP-ϵM is 1. Take M arbitrarily large but finite. Set $x_{41} > \frac{M-\epsilon}{2} + 3.5$. Given the value of M and the change made to x_{41}, all solutions with one misclassification are cut away now. Nevertheless, it is obvious that there still exist feasible solutions with only one misclassification provided the value of M would be sufficiently large. Pattern 4 is clearly a heavy outlier.

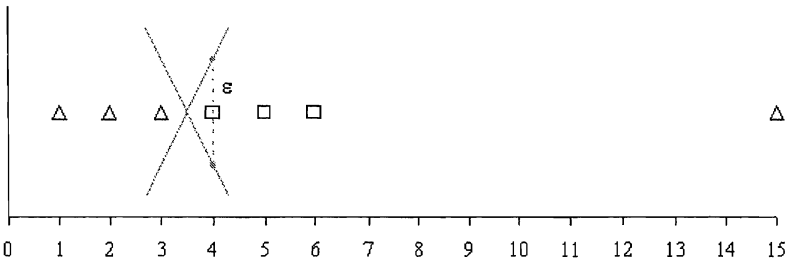


Figure 4: Example of how to construct an instance for which any finite M is too small.

C Appendix

The LP relaxation of MILP-ϵM has the trivial solution $a_{cp} = a_{dp}$ for $c, d \in \{1, ..., C\}$ and $p \in \{0, 1, ..., P\}$ if and only if there exist values $\breve{u}_{nc} \in \mathrm{IR}^+$ such that

$$\sum_{c=1}^{C} \breve{u}_{nc} = 1 \qquad n \in \{1, ..., N\}$$

$$N_c = \sum_{n=1\ c_n \neq c}^{N} \breve{u}_{nc} \qquad c \in \{1, ..., C\}$$

$$\sum_{n=1\ c_n=c}^{N} x_{np} = \sum_{n=1\ c_n \neq c}^{N} \breve{u}_{nc} x_{np} \qquad c \in \{1, ..., C\} \quad p \in \{1, ..., P\}.$$

Proof: The trivial solution is optimal if and only if its objective function value equals the objective function value of a dually feasible solution. Hence, the trivial solution is optimal if and only if there exist values for $u_{nc} \in \mathrm{IR}^+$ such that

$$\frac{N\epsilon}{M} = \epsilon(\sum_{n=1}^{N} \sum_{c=1}^{C} u_{nc}) \tag{1}$$

$$M(\sum_{c=1}^{C} u_{nc}) \leq 1 \qquad n \in \{1, ..., N\} \tag{2}$$

$$\sum_{n=1\ c_n=c}^{N} \sum_{c=1}^{C} u_{nc} - \sum_{n=1\ c_n \neq c}^{N} u_{nc} = 0 \qquad c \in \{1, ..., C\} \tag{3}$$

$$\sum_{n=1\ c_n=c}^{N} \sum_{c=1}^{C} u_{nc} x_{np} - \sum_{n=1\ c_n \neq c}^{N} u_{nc} x_{np} = 0 \qquad c \in \{1, ..., C\} \quad p \in \{1, ..., P\} \tag{4}$$

(1) is satisfied if and only if the N dual constraints in (2) are satisfied to equality. Using this and setting $\breve{u}_{nc} = u_{nc} M$, the optimality conditions for the trivial solution become

$$\sum_{c=1}^{C} \breve{u}_{nc} = 1 \qquad n \in \{1, ..., N\} \tag{5}$$

$$N_c = \sum_{n=1\ c_n \neq c}^{N} \breve{u}_{nc} \qquad c \in \{1, ..., C\} \tag{6}$$

$$\sum_{n=1\ c_n=c}^{N} x_{np} = \sum_{n=1\ c_n \neq c}^{N} \breve{u}_{nc} x_{np} \qquad c \in \{1, ..., C\} \quad p \in \{1, ..., P\} \tag{7}$$

where $\breve{u}_{nc} \in \mathrm{IR}^+$. (QED)

D Appendix

Table 6: SPEC marks found at http://www.spec.org

machine name	SPEC fp92	SPEC intfp92	SPEC fp95	SPEC intfp95
SPARCstation/server 20 Model 61 (60 MHz)	102.8	88.9	-	-
SPARCstation/server 5 (70 MHz)	47.3	57.0	-	-
SPARCstation 20 Model 151 (150 MHz)	-	-	4.71	4.02
SPARCstation 5 Model 170 (170 MHz)	-	-	3.00	3.53
Intel SE440BX2 Motherboard(550 MHz, Pentium III)	-	-	15.1	22.3

[14] reports CPU times in terms of DEC 3000 Alpha Model 300 (150 Mhz) seconds. We suggest the following crude approximation to compare their CPU times with ours. All CPU times in [14] are multiplied by 1.4 to obtain the equivalent CPU times on a SPARCstation/server 20 Model 61 (60 MHz). The factor of 1.4 is used by [14], based on 92 SPEC marks considerations. The more modern version of the SPARCstation/server 20 Model 61 (60 MHz) is the SPARCstation 20 Model 151 (150 MHz). It differs in CPU speed and memory. Assume that a task that takes one second on the 60 MHz machine takes 60/150 seconds on a 150 MHz computer. The 95 SPEC marks for the SPARCstation 20 Model 151 (150 MHz) and a machine similar to ours, the Intel SE440BX2 Motherboard (550 MHz, Pentium III), are given in 6. A task that takes one second on the SPARCstation 20 Model 151 (150 MHz) takes about 4.4 seconds on our machine. This gives the following scaling factor: $1.4 * 60/150 * 4.4 = 2.5$. The same steps can be repeated with the SPARCstation/server 5 (70 MHz) and its more modern counterpart, the SPARCstation 5 Model 170 (170 MHz). This gives a scaling factor of

$0.8 * 70/170 * 5.7 = 1.9$, which is of the same order of magnitude. The factor we will use is 2. The CPU times reported by [14] will be multiplied by two to allow for comparison with our CPU times.

References

[1] Bajgier S.M. and Hill A.V., 1982. An Experimental Comparison of Statistical and Linear Programming Approaches to the Discriminant Problem, Decision Sciences 13, pp. 604-618.

[2] Banks W.J. and Abad P.L., 1991. An Efficient Optimal Solution Algorithm for the Classification Problem, Decision Sciences 22, pp. 1008-1023.

[3] Bennett K.P. and Mangasarian O.L., 1993. Multicategory Discrimination via Linear Programming, Optimization Methods and Software 3, pp. 27-39.

[4] Bennet K.P. and Bredensteiner E.J., 1997. A Parametric Optimization Method for Machine Learning, INFORMS Journal on Computing, Volume 19, Number 3, pp. 311-318.

[5] Blake C.L. and Merz C.J., 1998. UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[6] Chinneck J.W., 2001. Fast Heuristics for the Maximum Feasible Subsystem Problem, INFORMS Journal on Computing, Volume 13, Number 3, pp. 210-223.

[7] Duda R.O., Hart P.E. and Strok D.H., 2001. Pattern Recognition, 2nd Edition. Wiley-Interscience, New York.

[8] Erenguc S.S. and Koehler G.J., 1990. Survey of Mathematical Programming Models and Experimental Results for Linear Discriminant Analysis, Managerial and Decision Economics, Volume 11, pp. 215-225.

[9] Gehrlein W.V., 1986. General Mathematical Programming Formulations for the Statistical Classification Problem, Operations Research Letters, Volume 5, Number 6, pp. 299-304.

[10] Gochet W., Stam A., Srinivasan V. and Chen S., 1997. Multigroup Discriminant Analysis Using Linear Programming, Operations Research, Volume 45, Number 2, pp. 213-225.

[11] Hand D.J., 1997. Construction and Assessment of Classification Rules. Wiley Chichester.

[12] Koehler G.J. and Erenguc S.S., 1990. Minimizing Misclassifications in Linear Discriminant Analysis, Decision Sciences 21, pp. 63-85.

[13] Liittschwager J.M. and Wang C., 1978. Integer Programming Solution of a Classification Problem, Management Science, Volume 24, Number 14, pp. 1515-1525.

[14] Lim T., Loh W. and Shih Y., 2000. A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms, Machine Learning, 40, pp. 203-229.

[15] Marcotte P., Marquis G. and Savard G., 1995. A New Implicit Enumeration Scheme for the Discriminant Analysis Problem, Computers Operations Research, Volume 22, Number 6, pp. 625-639.

[16] Rubin P.A., 1990. Heuristic Solution Procedures for a Mixed-Integer Programming Discriminant Model, Managerial and Decision Economics, Volume 11, pp. 255-266.

[17] Rubin P.A., 1997. Solving Mixed Integer Classification Problems by Decomposition, Annals of Operations Research 74, pp. 51-64.

[18] Schrage, L. 1995. LINDO: Optimization Software for Linear Programming. Lindo Systems Inc., Chicago, IL.

[19] Soltysik R.C. and Yarnold P.R., 1994. The Warmack-Gonzalez Algorithm for Linear Two-Category Multivariable Optimal Discriminant Analysis, Computers Operations Research, Volume 21, Number 7, pp. 735-745.

[20] Stam A. and Joachimsthaler E.A., 1990. A Comparison of a Robust Mixed-Integer Approach to Existing Methods for Establishing Classification Rules for the Discriminant Problem, European Journal of Operations Research 46 (1), pp. 113-122.

[21] Van Gestel T., Suykens J., Baesens B., Viaene S., Vanthienen J., Dedene G., De Moor B. and Vandewalle J., 2002. Benchmarking Least Squares Support Vector Machine Classifiers, Machine Learning, forthcoming.

[22] Webb A., 1999. Statistical Pattern Recognition. Arnold London.